SMBUD Project - Spark

Author(s): **Gabriele Ginestroni**
Giacomo Gumiero
Lorenzo Iovine
Nicola Landini
Francesco Leone

Group Number: **10**

Academic Year: 2022-2023

Contents

1 | Introduction

In this chapter will be presented the problem specification and the hypothesis under which the database is implemented.

2 | Conceptual Model

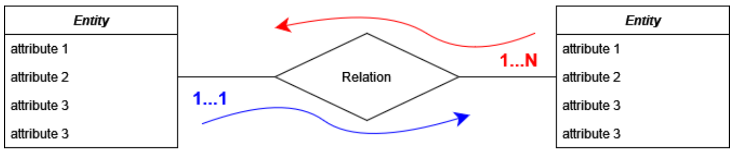


Figure 2.1: ER Diagram Organization

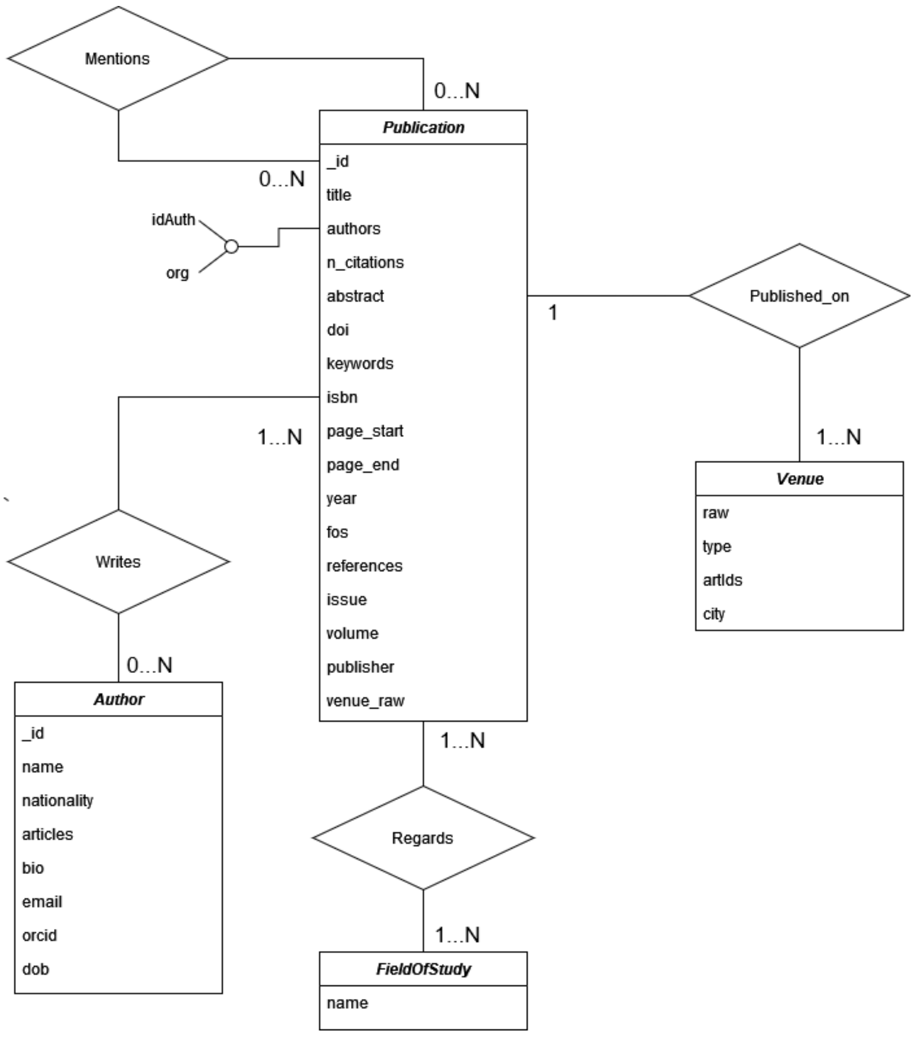


Figure 2.2: Conceptual Model

The conceptual model just showed contains 6 main entities, that are related to each other through various relationships:

- **Publication:** represents all the scientific articles. Its attributes are: *_id, title, authors, n_citations, abstract, doi, keywords, isbn, page_start, page_end, year, fos, references, issue, volume, publisher, venue_raw* and its organization will be presented later
- **Author:** it is the one who contributed to a publication. Its attributes are: *_id, name, nationality, articles, bio, email, orcid, dob (date of birth)*
- **Venue:** it is where a publication is published or presented. Its attributes are: *raw, type, artIds, city*
- **FieldOfStudy:** this entity represents the subjects of the publication and its attribute is *name*

The designed model contains also the following relationships:

- **Mentions:** is the relationship between a *publication* and another *publication* cited by the first one
- **Published_on:** is the relationship between a *publication* and its *venue*
- **Writes:** is the relationship between *author* and *publication* which features the affiliation property.
- **Regards:** is the relationship between a *publication* and its *fields of study*

3 | Data Structure

In this part of the project we used the same two `JSONs` used in the *MongoDB* implementation. We only removed chapters inside articles and `_id` authors inside articles was renamed as `idAuth`. Also id and dates were reconverted in plain text because in *MongoDB* we needed to add them as special entities: `$oid` and `$date`.
This was done via the following lines of script:

- the **first line** is used for the `JSON` file containing the articles. The script removes `$oid`, to delete chapters field and to rename authors field *id* into *idAuth*
- the **second line** is used for the `JSON` file containing the authors. The script removes `$oid` and `$date`

```
cat dblp_sample_filtered.json | sed -E 's/\{"\$oid":(["a-z0-9]+)\}/\1/g' | jq 'del(.[].chapters)' | jq '.[].authors[] |= with_entries(if .key == "id" then .key = "idAuth" else . end)' > dblp_sample_filtered_spark.json

cat dblp_sample_reverted_filtered.json | sed -E 's/\{"\$oid":(["a-z0-9]+)\}/\1/g' | sed -E 's/\{"\$date":(["A-Z0-9:-]+)\}/\1/g' > dblp_sample_reverted_filtered_spark.json
```

4 | Dataframe Structure

4.1. Article Structure

```
root
|-- _id: string (nullable = true)
|-- title: string (nullable = true)
|-- authors: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- idAuth: string (nullable = true)
|    |    |-- org: string (nullable = true)
|-- n_citation: integer (nullable = true)
|-- abstract: string (nullable = true)
|-- doi: string (nullable = true)
|-- keywords: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- isbn: string (nullable = true)
|-- page_start: string (nullable = true)
|-- page_end: string (nullable = true)
|-- year: integer (nullable = true)
|-- fos: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- references: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- issue: string (nullable = true)
|-- volume: string (nullable = true)
|-- publisher: string (nullable = true)
|-- venue_raw: string (nullable = true)
```

The structure just shown represents an *Article*; its attributes are:

- **_id** is the identifier of a publication.
- **title** represents the title of the publication.
- **authors** is an array that contains: `idAuth` of the authors of the article and the `org` field which represent the affiliation.

- **n_citation** is the number of times that the publication has been mentioned.
- **abstract** is a string containing a brief summary of the contents of the paper.
- **doi** Digital Object Identifier is a persistent and standardized identifier.
- **keywords** is an array containing keywords of the publication.
- **isbn** is an identification code of the venue of the publication.
- **page_start** defines the starting page of the publication.
- **page_end** defines the last page of the publication.
- **year** represents the year of publication.
- **fos** is an array containing the fields of study of the publication.
- **references** set of ObjectIds of the referenced articles.
- **issue** refers to how many times a periodical has been published during that year.
- **volume** is the volume of the venue in which the article has been published.
- **publisher** is the name of the publisher of the article.
- **venue_raw** is the name or the abbreviation of the venue (regardless the year, issue or volume) in which the publication was presented.

4.2. Author Structure

```
root
|-- _id: string (nullable = true)
|-- name: string (nullable = true)
|-- nationality: string (nullable = true)
|-- articles: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- bio: string (nullable = true)
|-- email: string (nullable = true)
|-- orcid: string (nullable = true)
|-- dob: timestamp (nullable = true)
```

The structure just shown represents an *Author*; its attributes are:

- **_id** is the identifier of an author.
- **name** is the name of the author.

- **nationality** is the nationality of the author.
- **articles** is a set of articles identifier of the publications of the author.
- **bio** is a string that describes the author.
- **email** is the email address of the author.
- **orcid** Open Researcher and Contributor ID is a unique identifier for authors of scientific articles.
- **dob** is the birth date of the author.

4.3. Venue Structure

```
root
|-- raw: string (nullable = true)
|-- type: integer (nullable = true)
|-- artIds: array (nullable = false)
|    |-- element: string (containsNull = false)
|-- city: string (nullable = true)
```

The structure just shown represents a *Venue*. This dataframe was obtained using data imported from the article `JSON` shown before. *Venue* attributes are the following:

- **raw** is the name or the abbreviation of the venue (regardless the year, issue or volume) in which the publication was presented.
- **type** indicates the type of the publication.
- **artIds** is a set of articles identifier associated to the venue.
- **city** represents the location of the venue an it is randomly populated.

5 Commands and Queries

5.1. Commands

We have identified the following `INSERT` and `UPDATE` commands to show the system basic functionalities.

5.1.1. Insert a new author

Assuming it is not present in the dataset, we created a new row with the values for the new author and we added it to the dataframe.

```python
new_author = Row(
    _id="638db170ae9ea0d19fad7a79",
    name="Emanuele Delle Valle ",
    nationality="it",
    articles=[],
    bio="Emanuele Della Valle holds a PhD in Computer Science from the \
        Vrije Universiteit Amsterdam and a Master degree in Computer Science\
        and Engineering from Politecnico di Milano. He is associate professor\
        at the Department of Electronics, Information and Bioengineering of\
        the Politecnico di Milano.",
    email="emanuele.dellavalle@gmail.com ",
    orcid="0000-0002-5176 -5885",
    dob= datetime.strptime("March 7, 1975", "%B %d, %Y")
)

df_authors = df_authors.union(spark.createDataFrame([new_author], schema = schemaAuthors))
```

5.1.2. Insert a new author

Assuming it is not present in the dataset, we created a new row with the values for a new article written by the author created in section 5.1.1. In order to set the authors, we instantiated an array `new_authors`.

```
new_authors =  [Row("638db170ae9ea0d19fad7a79", "Politecnico di Milano"), Row("638db170ae9ea0d19fad7a7a", "Politecnico di Milano")]

new_article = Row(
    _id="638db237d794b76f45c77916",
    title="An extensive study of C-SMOTE, a Continuous Synthetic Minority Oversampling Technique for Evolving Data Streams",
    authors=new_authors,
    n_citation=3,
    abstract = "Streaming Machine Learning (SML) studies algorithms that update their models,\
        given an unbounded and often non-stationary flow of data performing a single pass. Online \
        class imbalance learning is a branch of SML that combines the challenges of both class imbalance\
        and concept drift. In this paper, we investigate the binary classification problem by rebalancing\
        an imbalanced stream of data in the presence of concept drift, accessing one sample at a time.",
        doi="10.1016/j.eswa.2022.116630",
    keywords=["Evolving Data Stream","Streaming","Concept drift","Balancing"],
    isbn="123-4-567-89012-3",
    page_start="39",
    page_end="46",
    year=2022,
    fos=["Computer Science","Stream Reasoning","Big Data"],
    references=["53e99fe4b7602d97028bf743","53e99fddb7602d97028bc085"],
    issue="1",
    volume="196",
    publisher="Elsevier",
    venue_raw="ESA"
)

```

```
df_articles = df_articles.union(spark.createDataFrame([new_article]))
```

5.1.3. Insert a new venue

Assuming it is not present in the dataset, we created a new row with the values for a new venue *ESA* hosted in *Montreal*.
Note: in field `artIds` we set the article created in section 5.1.2.

```
new_venue = Row(
    raw="ESA",
    type=1,
    artIds=["638db237d794b76f45c77916"],
    city="Montreal"
)

df_venues = df_venues.union(spark.createDataFrame([new_venue]))
```

5.1.4. Insert a new article in his author dataframe

Through this command we inserted the article created in section 5.1.2 to its authors. In order to do that, we selected the authors through the ids and we add the article id to their field `articles`.
Note: one of the author is the one created in section 5.1.1.

```
df_authors = df_authors.withColumn(
    "articles",
    f.when(f.col("_id") == "638db170ae9ea0d19fad7a79",
        f.array_union(df_authors.articles, f.array(f.lit("
                                        638db237d794b76f45c77916"))))\
    .when(f.col("_id") == "638db170ae9ea0d19fad7a7a",
        f.array_union(df_authors.articles, f.array(f.lit("
                                        638db237d794b76f45c77916"))))
    .otherwise(f.col("articles"))
)
```

5.1.5. Update the number of citations of referenced publications

Through the following snippet of code is possible to increment the `n_citations` field of the *Publications* referenced by the article created in section 5.1.2.

Note: field `n_citations` is updated for both the referenced articles.

```
df_articles = df_articles.withColumn(
    "n_citation",
    f.when(f.col("_id") == "53e99fe4b7602d97028bf743",
        df_articles.n_citation+1) \
    .when(f.col("_id") == "53e99fddb7602d97028bc085",
        df_articles.n_citation+1)
    .otherwise(f.col("n_citation"))
)
```

5.1.6. Deleting an author from the database

Through the following snippet of code is possible to delete an author from the database. In order to do that we started from filtering on the identifier of the author to be removed and we deleted it.

```
df_authors.filter(f.col("_id") == "638db170ae9ea0d19fad7a79").show()
df_authors = df_authors.filter(f.col("_id") != "638db170ae9ea0d19fad7a79
                                    ")
```

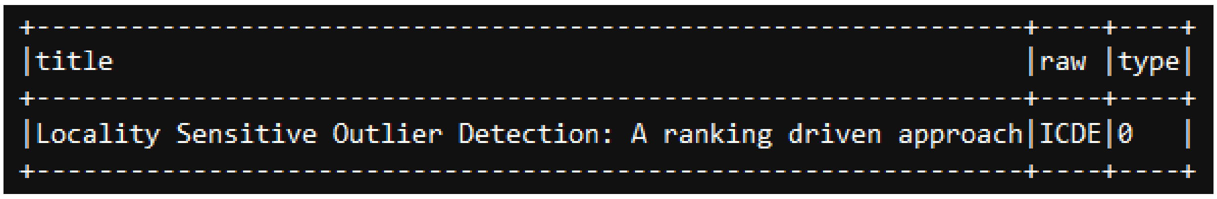
5.2. Queries

We have identified the following queries in order to show the system's basic functionalities. In the following sections title we wrote the basic requirements for every query, that, for ease of read, are represented as `SQL` clauses.

5.2.1. Query 1 - WHERE, JOIN

This query returns the type of the venue of an article with a the following title: *"Locality Sensitive Outlier Detection: A ranking driven approach"*.
Description: starting from the articles dataframe, a join is performed with the venues dataframe on the article's `venue_raw` field. After that, we filter the articles with the given title. Finally, we project over title,venue raw and venue type.

```
df_articles.join(df_venues, df_articles.venue_raw == df_venues.raw, "
                                    inner")\
          .filter(f.col("title") == "Locality Sensitive Outlier
                                            Detection: A ranking driven approach
                                            ")\
            .select("title", "raw", "type")\
            .show(truncate=False)
```

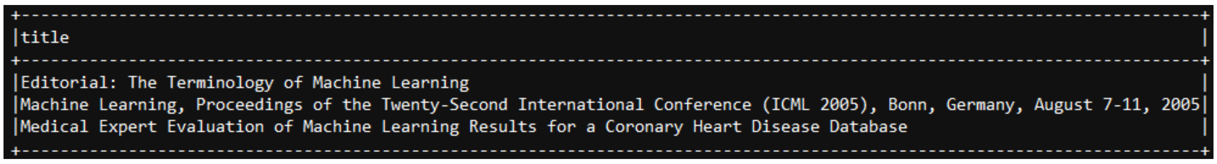


```
+---------------------------------------------------------------+----+----+
|title                                                          |raw |type|
+---------------------------------------------------------------+----+----+
|Locality Sensitive Outlier Detection: A ranking driven approach|ICDE|0   |
+---------------------------------------------------------------+----+----+
```

5.2.2. Query 2 - WHERE, LIMIT, LIKE

This query returns the articles whose title string contains *"Machine Learning"*.
Description: we filter the articles whose title contains "Machine Learning" using the like operator. Results are then limited to 3 tuples and projected over the article title.

```
df_articles.filter(f.col("title").like("%Machine Learning%"))\
            .limit(3)\
            .select("title")\
            .show(truncate=False)
```



```
+-------------------------------------------------------------------------------------------------------------------------+
|title                                                                                                                    |
+-------------------------------------------------------------------------------------------------------------------------+
|Editorial: The Terminology of Machine Learning                                                                           |
|Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005|
|Medical Expert Evaluation of Machine Learning Results for a Coronary Heart Disease Database                              |
+-------------------------------------------------------------------------------------------------------------------------+
```

5.2.3. Query 3 - WHERE, IN, NESTED QUERY

This query finds authors that has the same nationality of at least one of the authors of *"Locality Sensitive Outlier Detection: A ranking driven approach"* article.
Description: this query has been splitted in 2 queries:

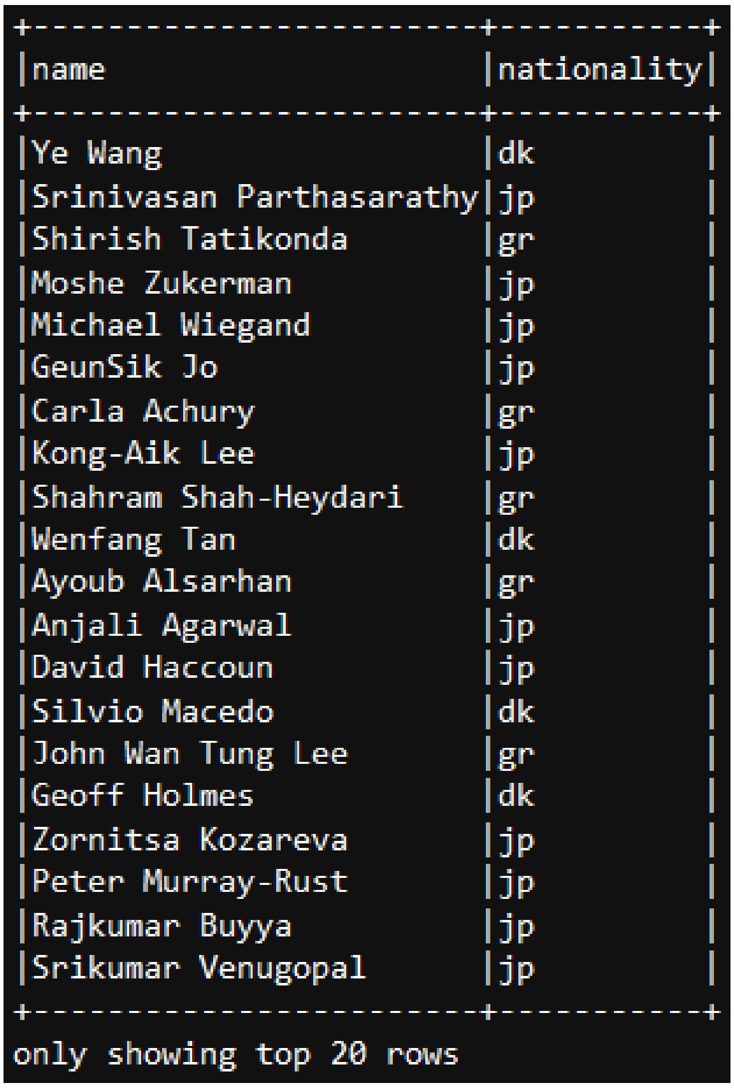
- *First query:* articles are filtered to find the article with the given title. After that, the authors array is exploded to perform a join on its idAuth field with the authors dataframe. Finally, nationalities of the article's authors are collected into a list using the `collect_set`.

`collect_set`, as the name suggests, discards duplicates, so the final list is a set of nationalities.

- *Second query:* starting from the authors' dataframe, we filtered all the authors whose nationality is present inside the list created with the previous query.

```
nationalities_list = df_articles.filter(f.col("title") == "Locality
                                    Sensitive Outlier Detection: A
                                    ranking driven approach")\
                                .select(f.explode(df_articles.authors.
                                    idAuth).alias("idAuth"))\
                                .join(df_authors, on=f.col("idAuth") ==
                                    df_authors._id)\
                                .select("nationality")\
                                .agg(f.collect_set("nationality")).
                                    collect()[0][0]

df_authors.filter(f.col("nationality")\
          .isin(nationalities_list))\
          .select("name","nationality")\
          .show(truncate=False)
```



```
+------------------------+-----------+
|name                    |nationality|
+------------------------+-----------+
|Ye Wang                 |dk         |
|Srinivasan Parthasarathy|jp         |
|Shirish Tatikonda       |gr         |
|Moshe Zukerman          |jp         |
|Michael Wiegand         |jp         |
|GeunSik Jo              |jp         |
|Carla Achury            |gr         |
|Kong-Aik Lee            |jp         |
|Shahram Shah-Heydari    |gr         |
|Wenfang Tan             |dk         |
|Ayoub Alsarhan          |gr         |
|Anjali Agarwal          |jp         |
|David Haccoun           |jp         |
|Silvio Macedo           |dk         |
|John Wan Tung Lee       |gr         |
|Geoff Holmes            |dk         |
|Zornitsa Kozareva       |jp         |
|Peter Murray-Rust       |jp         |
|Rajkumar Buyya          |jp         |
|Srikumar Venugopal      |jp         |
+------------------------+-----------+
only showing top 20 rows
```

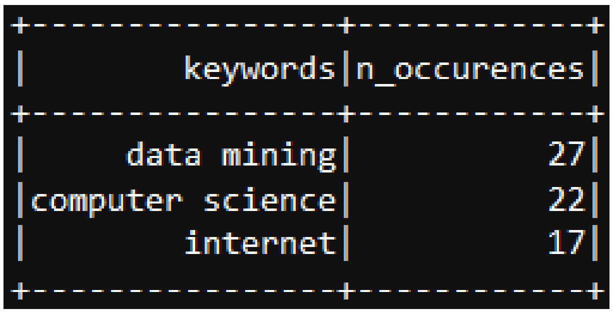
5.2.4. Query 4 - GROUP BY, JOIN, AS

This query finds the 3 most frequent keywords of articles written by italian authors.
Description: starting from the authors dataframe, we keep only italian authors and explode the articles field, renaming the new obtained field to `articles`. After that, duplicates are discarded.
In the second part of the query, we load the full articles's rows using a join. Then, keywords array is exploded. Keywords are grouped and counted. The groups are finally sorted and limited to show the top 3 keywords.

```
df_italian = df_authors.filter(f.col("nationality") == "it")\
                       .select(f.explode("articles")).withColumnRenamed(
                                   "col","articles")\
                       .distinct()
```

```

df_keywords = df_italian.join(df_articles, df_italian.articles ==
                                        df_articles._id, "inner")\
                          .select("articles", f.explode("keywords")).
                                        withColumnRenamed("col","keywords")\
                          .groupby("keywords")\
                          .agg(f.count("keywords").alias("n_occurences"))\
                          .sort("n_occurences", ascending=False)\
                          .limit(3).show()
```



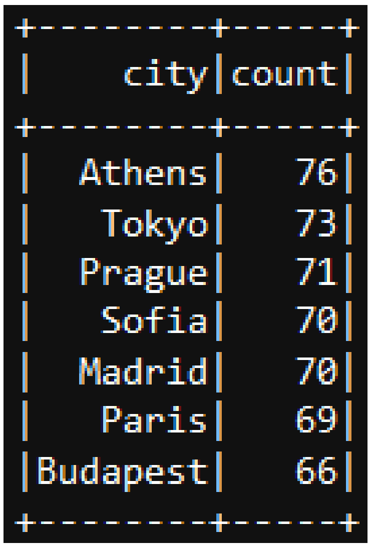
```
+----------------+------------+
|        keywords|n_occurences|
+----------------+------------+
|     data mining|          27|
|computer science|          22|
|        internet|          17|
+----------------+------------+
```

5.2.5. Query 5 - WHERE, GROUP BY

This query finds the cities with more than 65 venues.

Description: the venues dataframe is grouped with respect to the city to perform the count. After that, we keep only cities with more than 65 venues and sort the result in descending order.

```
df_venues\
    .groupby("city")\
    .count()\
    .filter(f.col("count") > 65)\
    .sort("count", ascending=False).show()
```

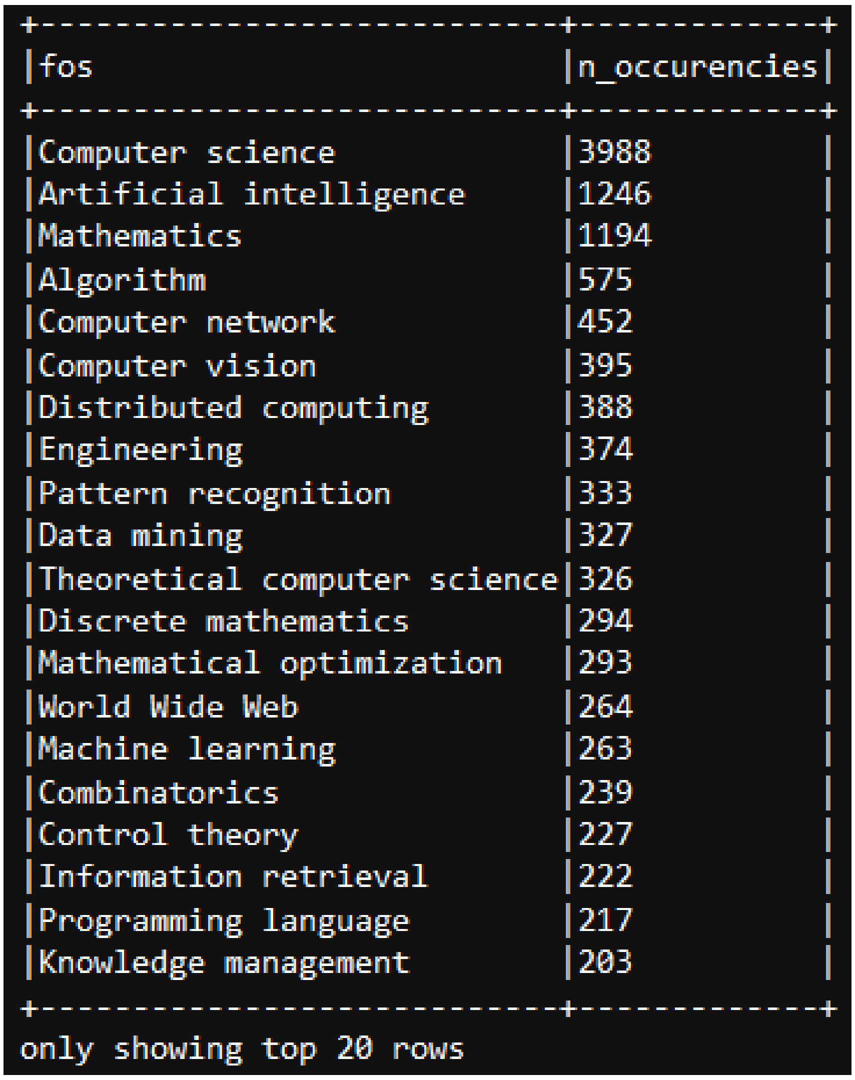


```
+--------+-----+
|    city|count|
+--------+-----+
|  Athens|   76|
|   Tokyo|   73|
|  Prague|   71|
|   Sofia|   70|
|  Madrid|   70|
|   Paris|   69|
|Budapest|   66|
+--------+-----+
```

5.2.6. Query 6 - GROUP BY, HAVING, AS

This query finds the field of studies that appears more than 15 times.
Description: We use the explode function to convert the fos array into multiple rows,then we rename the resulting column to `fos`, group by `fos` and count the number of occurrences.
After that, we keep rows with more than 15 occurrences, sort the remaining rows in descending order based on the number of occurrences, and show the top rows.

```
df_articles\
    .select("_id", "title", f.explode("fos")).withColumnRenamed("col", "
        fos")\
    .groupby("fos")\
    .agg(f.count("fos").alias("n_occurencies"))\
    .filter(f.col("n_occurencies") > 15)\
    .sort("n_occurencies", ascending=False)\
    .show(truncate=False)
```



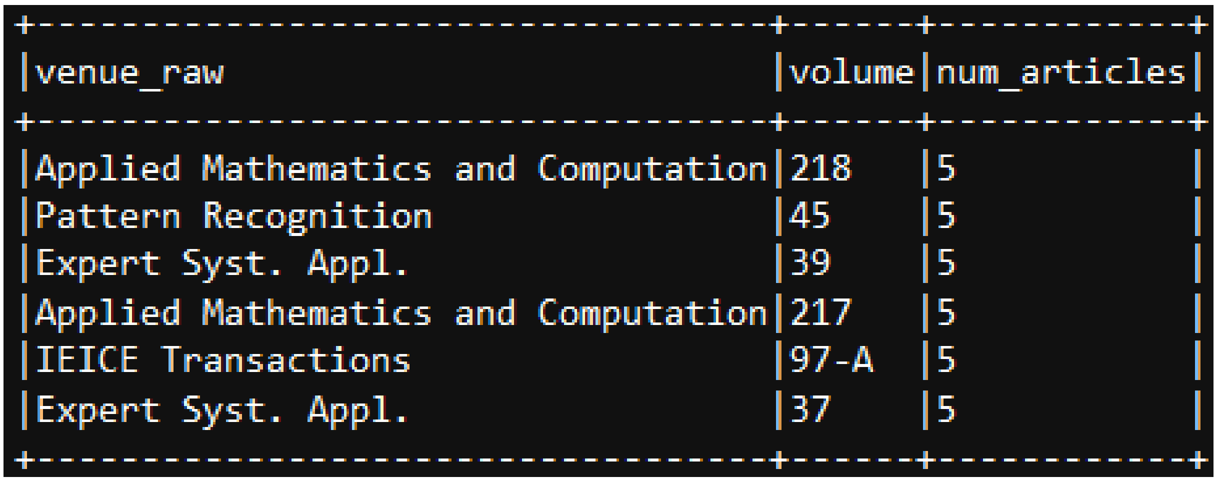
```
+----------------------------+-------------+
|fos                         |n_occurencies|
+----------------------------+-------------+
|Computer science            |3988         |
|Artificial intelligence     |1246         |
|Mathematics                 |1194         |
|Algorithm                   |575          |
|Computer network            |452          |
|Computer vision             |395          |
|Distributed computing       |388          |
|Engineering                 |374          |
|Pattern recognition         |333          |
|Data mining                 |327          |
|Theoretical computer science|326          |
|Discrete mathematics        |294          |
|Mathematical optimization   |293          |
|World Wide Web              |264          |
|Machine learning            |263          |
|Combinatorics               |239          |
|Control theory              |227          |
|Information retrieval       |222          |
|Programming language        |217          |
|Knowledge management        |203          |
+----------------------------+-------------+
only showing top 20 rows
```

5.2.7. Query 7 - WHERE, GROUP BY, HAVING, AS

This query finds all the volumes with at least 5 articles in the dataset, published after 2000.

Description: This query filters the articles in the articles dataframe to only those published after the year 2000, then groups the remaining articles by `venue_raw` and `volume`, counts the number of articles per group, filters the groups to only those with more than 4 articles, and finally displays the results.

```
df_articles\
    .filter(f.col("year") > 2000)\
    .groupby("venue_raw", "volume")\
    .agg(f.count("volume").alias("num_articles"))\
    .filter(f.col("num_articles") > 4)\
    .show(truncate = False)
```



```
+------------------------------------+------+------------+
|venue_raw                           |volume|num_articles|
+------------------------------------+------+------------+
|Applied Mathematics and Computation|218   |5           |
|Pattern Recognition                 |45    |5           |
|Expert Syst. Appl.                  |39    |5           |
|Applied Mathematics and Computation|217   |5           |
|IEICE Transactions                  |97-A  |5           |
|Expert Syst. Appl.                  |37    |5           |
+------------------------------------+------+------------+
```

5.2.8. Query 8 - WHERE, NESTED QUERY, GROUP BY

The following query is divided in two queries:

8a. find the venue with highest number of articles

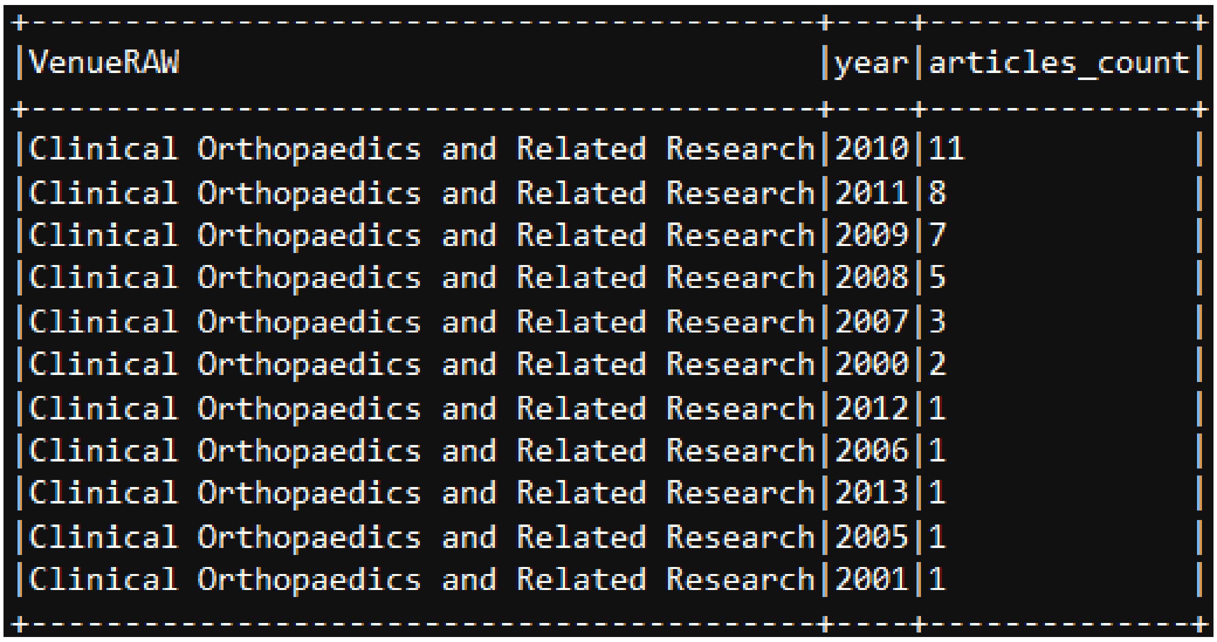
8b. find the number of articles published per year on that venue

Description: the basic functionalities of the two queries are the following

- the *first query* selects the top venue from the venue dataframe based on the size of the `artIds` attribute
- the *second query* filters articles with the selected `venue_raw`, groups the articles by year, and counts the number of articles in each group. Finally, it displays the results projecting over `top_venue`, `year` and `articles_count`

```
top_venue = df_venues\
                .select("raw",f.size("artIds").alias("count"))\
                .orderBy("count",ascending = False)\
                .limit(1)

df_articles_year = df_articles\
                        .filter(f.col("venue_raw") == top_venue.collect
                            ()[0][0])\
                        .groupBy("year")\
                        .count()\
                                .orderBy("count",ascending=False)\
                                .select(f.lit(top_venue.collect()[0][0]).alias("
                                              VenueRAW"),"year",f.col("count").
                                              alias("articles_count"))\
                                .show(truncate=False)
```



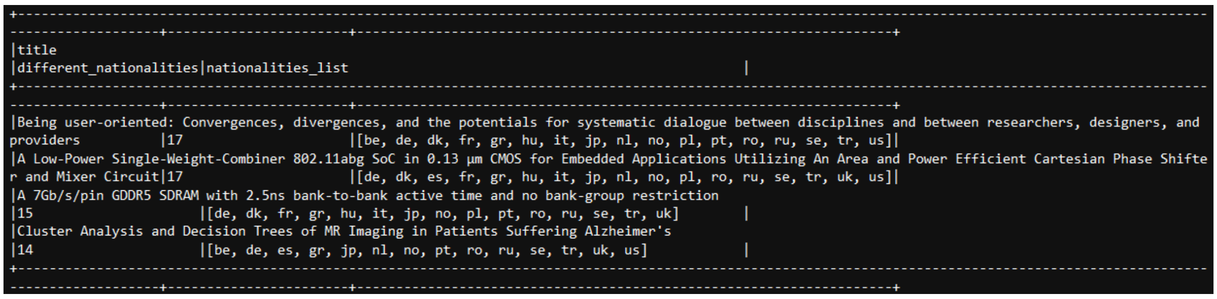
```
+----------------------------------------+----+--------------+
|VenueRAW                                |year|articles_count|
+----------------------------------------+----+--------------+
|Clinical Orthopaedics and Related Research|2010|11            |
|Clinical Orthopaedics and Related Research|2011|8             |
|Clinical Orthopaedics and Related Research|2009|7             |
|Clinical Orthopaedics and Related Research|2008|5             |
|Clinical Orthopaedics and Related Research|2007|3             |
|Clinical Orthopaedics and Related Research|2000|2             |
|Clinical Orthopaedics and Related Research|2012|1             |
|Clinical Orthopaedics and Related Research|2006|1             |
|Clinical Orthopaedics and Related Research|2013|1             |
|Clinical Orthopaedics and Related Research|2005|1             |
|Clinical Orthopaedics and Related Research|2001|1             |
+----------------------------------------+----+--------------+
```

5.2.9. Query 9 - WHERE, GROUP BY, HAVING, 1 JOIN

The following query finds the articles, published after 2000, with more than 13 different nationalities of its authors.

Description: this query filters the articles dataframe by year, exploding the authors array `idAuth` field (note that authors is an array of struct elements with 2 fields). After that, it joins the result with the authors dataframe, groups the articles by id and counts the number of distinct nationalities among the authors. It finally filters the results to include only articles with more than 13 different nationalities and orders the results in descending order, displaying the title of the article, the list of nationalities and their count.

```
df_articles_nationalities = df_articles.alias("art")\
                                .filter(f.col("year") > 2000)\
                                .select("art._id","art.title", f.explode
                                    ("art.authors.idAuth").alias("author
                                    "))\
                                        .join(df_authors.alias("auth"), on=f.col
                                            ("author") == df_authors._id)\
                                        .groupBy("art._id")\
                                        .agg(f.first("title").alias("title"),f.
                                            countDistinct("nationality").alias("
                                            different_nationalities"), f.
                                            collect_set("nationality").alias("
                                            nationalities_list"))\
                                        .filter(f.col("different_nationalities")
                                             > 13)\
                                        .orderBy("different_nationalities",
                                            ascending=False)\
                                        .select("title","different_nationalities
                                            ",f.sort_array("nationalities_list")
                                            .alias("nationalities_list"))\
                                        .show(truncate=False)
```



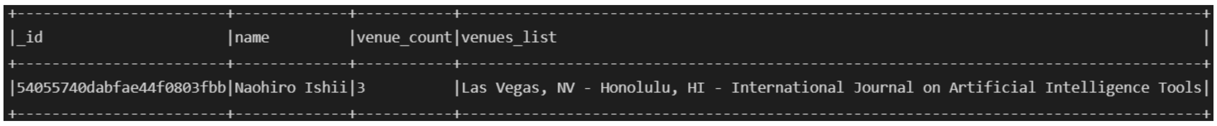
```
+------------------------------------------------------------------------------------------------------------------------------------
------------------+---------------------+------------------------------------------------------------+
|title
|different_nationalities|nationalities_list                                          |
+------------------------------------------------------------------------------------------------------------------------------------
------------------+---------------------+------------------------------------------------------------+
|Being user-oriented: Convergences, divergences, and the potentials for systematic dialogue between disciplines and between researchers, designers, and
providers         |17                   |[be, de, dk, fr, gr, hu, it, jp, nl, no, pl, pt, ro, ru, se, tr, us]|
|A Low-Power Single-Weight-Combiner 802.11abg SoC in 0.13 µm CMOS for Embedded Applications Utilizing An Area and Power Efficient Cartesian Phase Shifte
r and Mixer Circuit|17                   |[de, dk, es, fr, gr, hu, it, jp, nl, no, pl, ro, ru, se, tr, uk, us]|
|A 7Gb/s/pin GDDR5 SDRAM with 2.5ns bank-to-bank active time and no bank-group restriction
|15                     |[de, dk, fr, gr, hu, it, jp, no, pl, pt, ro, ru, se, tr, uk]          |
|Cluster Analysis and Decision Trees of MR Imaging in Patients Suffering Alzheimer's
|14                     |[be, de, es, gr, jp, nl, no, pt, ro, ru, se, tr, uk, us]              |
+------------------------------------------------------------------------------------------------------------------------------------
------------------+---------------------+------------------------------------------------------------+
```

5.2.10. Query 10 - WHERE, GROUP BY, HAVING, 2 JOINS

This query finds all the authors that published on more than 2 Journals.

Description: starting from the authors dataframe, we explode the articles array, creating a new field named `article`. After that, we join the results with the articles collection and then with the venues collection. Then, we filter the results to keep only articles written on journals (type 1) and group by the author id. Finally, we count the number of distinct venues in each group (collecting in a list all the venues of the group), and keep only the groups with more than 2 venues.

```
df_exploded_authors = df_authors.alias("auth")\
                        .select("auth._id","auth.name", f.explode("auth.
                                    articles").alias("article"))\
                        .join(df_articles.alias("art"), on=f.col("
                                    article") == df_articles._id)\
                                    .select("auth._id","auth.name","art._id","art.venue_raw")\
                                    .join(df_venues.alias("ven"), on=f.col("venue_raw") == df_venues.raw)\
                                    .filter(f.col("type") == 1)\
                                    .groupBy("auth._id")\
                                    .agg(f.first("name").alias("name"),f.countDistinct("raw").alias("venue_count"),f.concat_ws(" - ",f.collect_set("raw")).alias("venues_list"))\
                                    .filter(f.col("venue_count") > 2)\
                                    .orderBy("venue_count", ascending=False).show(3, truncate=False)
```



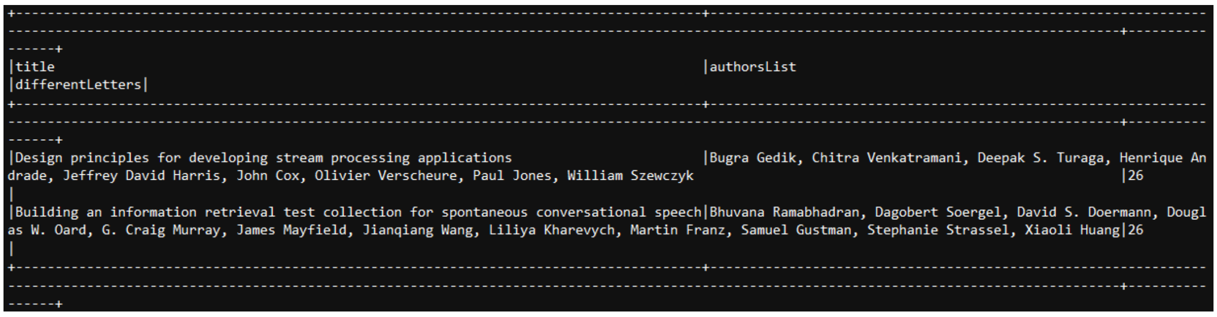
```
+------------------------+-------------+-----------+-------------------------------------------------------------------------------------+
|_id                     |name         |venue_count|venues_list                                                                          |
+------------------------+-------------+-----------+-------------------------------------------------------------------------------------+
|54055740dabfae44f0803fbb|Naohiro Ishii|3          |Las Vegas, NV - Honolulu, HI - International Journal on Artificial Intelligence Tools|
+------------------------+-------------+-----------+-------------------------------------------------------------------------------------+
```

5.2.11. Query 11 - EXTRA

This query returns all the articles written by authors whose names combined have all 26 letters of the alphabet.

Description: The query starts with exploding articles for each author. Then, the grouping combined with the collect retrieves, for each article, the list of its authors, then several operation are applied on this list in order to obtain the different letters that are present in the list of authors. After that, a filter to keep only the ones that have all the 26 letters of the alphabet in it is applied, and the result is joined with articles to obtain the title. Finally, a projection is used to display the title and the list of authors in alphabetical order.

```
df_authors.select("name", f.explode("articles").alias("idArt")) \
            .groupBy("idArt") \
            .agg(f.collect_set("name").alias("authorsList")) \
            .select("idArt", "authorsList", (f.size(f.array_distinct(f.split(f.regexp_replace(f.lower(f.concat_ws("", "authorsList")), "[^a-z]", ""), "")))-1).alias("differentLetters")) \
            .filter(f.col("differentLetters") == 26)\
            .join(df_articles, on=f.col("idArt") == df_articles._id) \
            .select("title", f.concat_ws(", ", f.sort_array("authorsList")
                                         ).alias("authorsList"), "
                                         differentLetters") \
            .show(truncate=False)
```

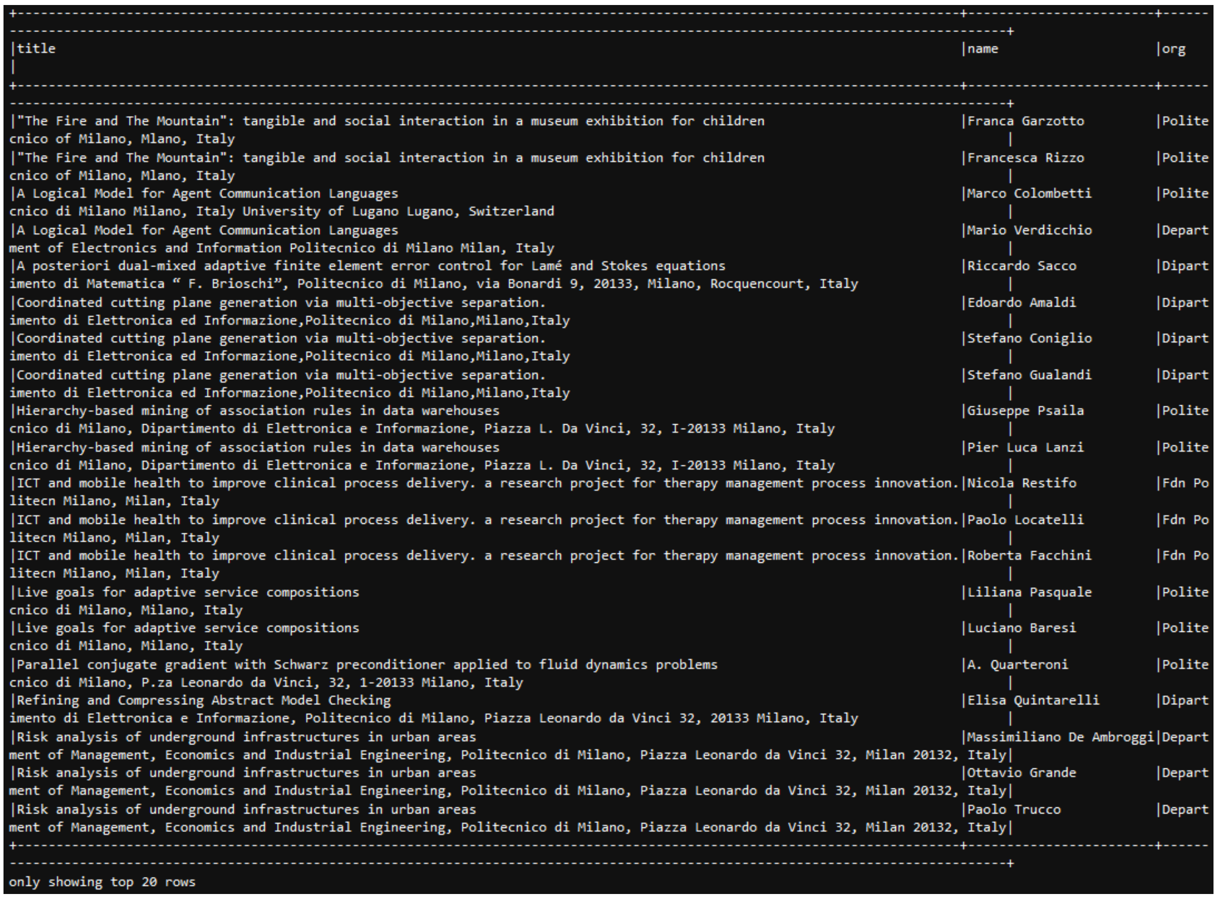


```
+---------------------------------------------------------------------------------------+---------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------+----------
------+
|title                                                                                  |authorsList
|differentLetters|
+---------------------------------------------------------------------------------------+---------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------+----------
------+
|Design principles for developing stream processing applications                        |Bugra Gedik, Chitra Venkatramani, Deepak S. Turaga, Henrique An
drade, Jeffrey David Harris, John Cox, Olivier Verscheure, Paul Jones, William Szewczyk                                                      |26
|
|Building an information retrieval test collection for spontaneous conversational speech|Bhuvana Ramabhadran, Dagobert Soergel, David S. Doermann, Dougl
as W. Oard, G. Craig Murray, James Mayfield, Jianqiang Wang, Liliya Kharevych, Martin Franz, Samuel Gustman, Stephanie Strassel, Xiaoli Huang|26
|
+---------------------------------------------------------------------------------------+---------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------+----------
------+
```

5.2.12. Query 12 - EXTRA

This query returns all articles written in affiliation with *Politecnico of Milano*.
Description: the query starts by exploding the authors array field in the article, creating the new `affiliation` attribute. Articles that contain at least one of the desired organization (the same article could be written in collaboration with different universities) are kept. Then, a join with the authors collection is executed to retrieve the name of the author.

```
df_articles\
    .select("title",f.explode("authors").alias("affiliation"))\
    .filter(f.col("affiliation.org").like("%Poli%Mil%"))\
    .join(df_authors, on=f.col("affiliation.idAuth") == df_authors._id)
                                                   \
    .select("title", "name", "affiliation.org") \
    .orderBy("title","name")\
    .show(truncate=False)
```



```
+-------------------------------------------------------------------------------------------------------------------------+------------------------+------
--------------------------------------------------------------------------------------------------------------------------------+
|title                                                                                                                    |name                    |org
|
+-------------------------------------------------------------------------------------------------------------------------+------------------------+------
--------------------------------------------------------------------------------------------------------------------------------+
|"The Fire and The Mountain": tangible and social interaction in a museum exhibition for children                         |Franca Garzotto         |Polite
cnico of Milano, Mlano, Italy                                                                                                   |
|"The Fire and The Mountain": tangible and social interaction in a museum exhibition for children                         |Francesca Rizzo         |Polite
cnico of Milano, Mlano, Italy                                                                                                   |
|A Logical Model for Agent Communication Languages                                                                        |Marco Colombetti        |Polite
cnico di Milano Milano, Italy University of Lugano Lugano, Switzerland                                                          |
|A Logical Model for Agent Communication Languages                                                                        |Mario Verdicchio        |Depart
ment of Electronics and Information Politecnico di Milano Milan, Italy                                                          |
|A posteriori dual-mixed adaptive finite element error control for Lamé and Stokes equations                              |Riccardo Sacco          |Dipart
imento di Matematica “ F. Brioschi”, Politecnico di Milano, via Bonardi 9, 20133, Milano, Rocquencourt, Italy                   |
|Coordinated cutting plane generation via multi-objective separation.                                                     |Edoardo Amaldi          |Dipart
imento di Elettronica ed Informazione,Politecnico di Milano,Milano,Italy                                                        |
|Coordinated cutting plane generation via multi-objective separation.                                                     |Stefano Coniglio        |Dipart
imento di Elettronica ed Informazione,Politecnico di Milano,Milano,Italy                                                        |
|Coordinated cutting plane generation via multi-objective separation.                                                     |Stefano Gualandi        |Dipart
imento di Elettronica ed Informazione,Politecnico di Milano,Milano,Italy                                                        |
|Hierarchy-based mining of association rules in data warehouses                                                          |Giuseppe Psaila         |Polite
cnico di Milano, Dipartimento di Elettronica e Informazione, Piazza L. Da Vinci, 32, I-20133 Milano, Italy                      |
|Hierarchy-based mining of association rules in data warehouses                                                          |Pier Luca Lanzi         |Polite
cnico di Milano, Dipartimento di Elettronica e Informazione, Piazza L. Da Vinci, 32, I-20133 Milano, Italy                      |
|ICT and mobile health to improve clinical process delivery. a research project for therapy management process innovation.|Nicola Restifo          |Fdn Po
litecn Milano, Milan, Italy                                                                                                     |
|ICT and mobile health to improve clinical process delivery. a research project for therapy management process innovation.|Paolo Locatelli         |Fdn Po
litecn Milano, Milan, Italy                                                                                                     |
|ICT and mobile health to improve clinical process delivery. a research project for therapy management process innovation.|Roberta Facchini        |Fdn Po
litecn Milano, Milan, Italy                                                                                                     |
|Live goals for adaptive service compositions                                                                             |Liliana Pasquale        |Polite
cnico di Milano, Milano, Italy                                                                                                  |
|Live goals for adaptive service compositions                                                                             |Luciano Baresi          |Polite
cnico di Milano, Milano, Italy                                                                                                  |
|Parallel conjugate gradient with Schwarz preconditioner applied to fluid dynamics problems                               |A. Quarteroni           |Polite
cnico di Milano, P.za Leonardo da Vinci, 32, 1-20133 Milano, Italy                                                              |
|Refining and Compressing Abstract Model Checking                                                                         |Elisa Quintarelli       |Dipart
imento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy                   |
|Risk analysis of underground infrastructures in urban areas                                                              |Massimiliano De Ambroggi|Depart
ment of Management, Economics and Industrial Engineering, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milan 20132, Italy|
|Risk analysis of underground infrastructures in urban areas                                                              |Ottavio Grande          |Depart
ment of Management, Economics and Industrial Engineering, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milan 20132, Italy|
|Risk analysis of underground infrastructures in urban areas                                                              |Paolo Trucco            |Depart
ment of Management, Economics and Industrial Engineering, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milan 20132, Italy|
+-------------------------------------------------------------------------------------------------------------------------+------------------------+------
--------------------------------------------------------------------------------------------------------------------------------+
only showing top 20 rows
```